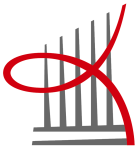


Työmaan reaaliaikainen kommunikaatiojärjestelmä käyttäen XMPP-protokollaa

Oskari Timperi

Opinnäytetyö
Joulukuu 2011
Tietotekniikka
Ohjelmistotekniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

TIMPERI, OSKARI: Työmaan reaaliaikainen kommunikaatiojärjestelmä käyttäen XMPP-protokollaa

Opinnäytetyö 30 s., liitteet 11 s.
Joulukuu 2011

Nykyään yksi rakennushankkeiden ongelmista on automaattisen tiedonkeruun puutteellisuus tai puuttuminen kokonaan. Hankkeissa voi olla käytössä useita järjestelmiä, jotka eivät osaa keskustella eivätkä jakaa tietoa keskenään. Osa rakennushankkeiden dokumentaatiosta on paperisena ja osa sähköisenä eri formaateissa. Tämä hankaloittaa huomattavasti järjestelmien yhteistoimintaa. Dokumentaation tulisi olla kaikkien ymmärtämässä sähköisessä muodossa, jolloin sen jakaminenkin olisi yksinkertaisempaa.

Tämän opinnäytetyön tavoitteena oli suunnitella kommunikointiprotokolla sekä kehittää sitä käyttävä järjestelmä, jonka avulla muut järjestelmät voivat vaihtaa keskenään helposti tietoa. Protokollan tulee myös olla helppokäyttöinen, laajennettava sekä avoin.

Opinnäytetyön puitteissa suunniteltu protokolla sisältää määrittelyn tilatiedon ja paikkatiedon jakamiseen työmaalla. Protokolla on täysin avoin ja helposti laajennettavissa moneen käyttökohteeseen. Protokollan suunnittelun lisäksi toteutettiin alustava järjestelmä, joka integroidaan myöhempänä ajankohtana Novatron Oy:n laitteisiin.

Asiasanat: tiedonsiirto, maanrakennus, koneohjaus

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Software Engineering

TIMPERI, OSKARI: Real-time Communication System for Worksites Using XMPP Protocol

Bachelor's thesis 30 pages, appendices 11 pages
December 2011

One of the main problems in construction projects today is the inadequate use or lack of automatic data acquisition. Projects may have various data systems that cannot exchange information between each other. Part of the project documentation may be in paper format and some other part may be in different electronic formats. This makes it hard for different systems to communicate.

The purpose of this thesis was to design a communication protocol that could be used between different systems to easily exchange information. The protocol should be easy to use, extensible and open. Also a preliminary system using the protocol would be needed.

The protocol designed in the limits of this thesis includes specifications for exchanging status and location information. The protocol is fully open and extensible to many uses. In addition to the protocol a system was developed that uses the protocol. The system is integrated to Novatron Oy's products at a later time.

Keywords: information exchange, construction, machine control

Esipuhe

Tämän opinnäytetyön tekeminen on ollut melkoinen prosessi. Aihetta ehdotettiin minulle jo toukokuussa 2011, jolloin olin työskennellyt Novatronilla vajaan kuukauden. Koneenohjaus ja maanrakennus olivat kohtuullisen uusia asioita. Opinnäytetyön tekemisen aloitin syksyllä. Varsinainen tekstin kirjoittaminen alkoi lokakuussa.

Aikataulun sovittaminen opinnäytetyölle, koululle, järjestötoiminnalle sekä kaikelle muulle oli varsin haastavaa ja nyt tuntuukin, että tunnelin päässä näkyy valoa.

Haluaisin kiittää työn valvojaa Jari Mikkolaista rennosta otteesta, kolleegaani Jarko Leppästä hyvistä ideoista ja neuvoista, esimiestäni Visa Hokkasta haastavasta aiheesta, muuta Novatronin henkilöstöä mielenkiintoisesta työympäristöstä, sekä Leena Koskimäkeä tuesta syksyn aikana.

Tampereella joulukuussa 2011

Oskari Timperi

Sisältö

Sanasto

Base64	Binääritiedon koodaustapa, joka perustuu lukujärjestelmään jonka kantaluku on 64
Binääri-XML	Mikä tahansa määrittely joka määrittää XML:lle tiiviimmän esitystavan binäärimuodossa
Botti	Sanasta *robotti*; sovellus joka toimii itsenäisesti sille määriteltyjen toimintaohjeiden puitteissa
DMUC	Distributed MUC; hajautettu MUC -palvelu
EXI	Efficient XML Interchange; eräs binääri-XML formaatti
GPRS	General Packet Radio Service
GZIP	GNU zip; Tiedostonpakkausohjelma sekä tiedostoformaatti
HTML	Hyper Text Markup Language
IETF	The Internet Engineering Task Force; yhteisö joka edesauttaa Internet-tekniikoiden kehittämistä
IRC	Internet Relay Chat
Jabber	XMPP:n alkuperäinen nimi
JID	Jabber Identifier; XMPP -osoitteen nimitys
LandXML	XML-pohjainen tiedonsiirtoformaatti suunnittelu- ja mittausjärjestelmäväelle
MUC	Multi User Chat; XMPP:ssä käytettävän ryhmäkeskusteluprotokollan nimi
OgreMesh	Ogre3D:ssä käytettävä XML-esitys malleille
PubSub	Publish & Subscribe
UCS	Universal Character Set; yleismaailmallinen merkistö joka sisältää noin 100000 merkkiä
Unicode	Merkistöstandardi joka sisältää tekstin koodauksen, esityksen sekä käsittelyn suurimmalle osalle maailman kirjoitusjärjestelmistä
UTF-8	UCS Transformation Format - 8-bit; ASCIIn kanssa yhteensopiva merkistökoodaus jolla voidaan esittää kaikki merkit UCS -merkistöstä
VoIP	Voice over IP
W3C	World Wide Web Consortium; kansainvälinen yritysten ja yhteisöjen yhteenliittymä joka ylläpitää WWW:n standardeja
XEP	XMPP Extension Protocol; XMPP -protokollan laajennos
XML	Extensible Markup Language
XML-skeema	XML-dokumenttien rakenteen kuvaamiseen kehitetty teknologia

XML-virta	engl. XML stream; esimerkiksi verkosta luettava ja samalla tulkittava XML -dokumentti
XMPP	Extensible Messaging and Presence Protocol
XSF	XMPP Standards Foundation

1 Johdanto

1.1 Tausta

Tällä hetkellä yhdeksi rakennusprojektien ongelmaksi koetaan automaattinen tiedonkeruu työmaalla. Käytössä voi olla useita erilaisia järjestelmiä ja niiden integroiminen on ongelmallista. Rakennusprojektiin liittyviä järjestelmiä voivat olla esimerkiksi koneenohjausjärjestelmät, työmaan suunnitteluohjelmistot ja tiedonkeruuseen tarkoitetut tietopankit. Standardoitua tapaa järjestelmien väliseen kommunikointiin ei ole olemassa.

Tiedonkeruu varsinaisen rakentamisen aikana on vain yksi osa kaikkea projektiin liittyvää tietoa. Työmaalla tapahtuvaan tiedonkeruuseen liittyy kiinteästi suunnittelun aikana tehdyt suunnitelmat ja rakentamisen jälkeen suoritettava työn todentaminen. Suunnitelmien perusteella voidaan luoda koneenohjaukselle mallit. Työn aikana kerättyä tietoa voidaan verrata suunnitelmiin ja sitä voidaan käyttää myös todentamiseen. Kaikkien näiden vaiheiden, kuten rakentaminen, suunnittelu ja todentaminen, välillä tapahtuu siis tiedonsiirtoa.

On todettu, että rakennushankkeiden vaiheiden välisessä tiedonsiirrossa on parantamisen varaa (Kilpeläinen 2007; Infra - Rakentaminen ja palvelut 2001-2005). Suuri osa projektiin liittyvästä tiedosta on sähköisessä muodossa. Toisinaan käytetään kuitenkin paperisia dokumentteja. Sekalaisen tiedon yhdistäminen ja ylläpito on vaikeaa. Tietotekniikan yleistyessä myös rakentamisessa on pyritty hyödyntämään digitaalisia tietomalleja. Tietomalli on projektin koko elinkaaren aikaisen tiedon kokonaisuus digitaalisessa muodossa.

Kuva 1: Rakennusprojektin elinkaari ja siihen liittyvät mallit (InfraTM TEKES loppuraportti 2010)

Tulevaisuudessa projektien kaikki dokumentaatio ja materiaali tulisi olla digitaalisessa muodossa. Teoriassa järjestelmien välisen kommunikaation tulisi toimia saumattomasti. Suunnittelijan jollain työkalulla tehdyt työmaasuunnitelmat (geometria kolmiulotteisesti) ja muu dokumentaatio voidaan ladata koneenohjausjärjestelmään, joka pystyy suunnitelmien perusteella opastamaan työkoneiden käyttäjiä tai toimimaan jopa autonomisesti.

1.2 Tavoite

Tämän opinnäytetyön tavoitteena on määritellä kommunikointiprotokolla sekä -järjestelmä, jota voidaan käyttää automaattisessa tiedonkeruussa sekä yleisesti työmaan kommunikoinnissa. Järjestelmän tulisi helpottaa suunnitelmien siirtämistä koneenohjaukseen ja tehdyn työn todentamista.

Tämä opinnäytetyö keskittyy pääasiassa tiedonkeruuseen. Tiedonkeruun osalta keskitytään tila- ja paikkatietoon. Tilatiedolla tarkoitetaan tietoa, joka kertoo mitä esimerkiksi työkone tekee tietyllä hetkellä. Paikkatieto kertoo, missä koneet liikkuvat. Tila- ja paikkatiedon avulla voidaan jo tehdä monenlaisia sovelluksia, kuten tehdyn työn määrän seurantaa.

Järjestelmän suunnittelun lähtökohtina on ollut yksinkertaisuus, laajennettavuus ja avoimuus. Järjestelmän tulee olla helposti laajennettavissa. Laajennettavuus takaa sen, että siihen on jälkikäteen helppoa ja vaivatonta lisätä uusia ominaisuuksia, ilman että vanhat ominaisuudet muuttuisivat radikaalisti.

Järjestelmän täytyy myös olla avoin siinä merkityksessä, että järjestelmän määrittelyt ovat ilmaisia. Kuka tahansa voi toteuttaa järjestelmän, kunhan se pystyy kommunikoimaan muiden vastaavien järjestelmien kanssa tässä työssä määritellyllä protokollalla. Avoimuus on avainasemassa järjestelmän käyttöönoton kannalta.

Kun työmaalta voidaan helposti kerätä tietoa eri tietopankkeihin, pystytään työmaan etenemistä seuraamaan reaaliajassa. Järjestelmästä on eniten hyötyä työnjohdolle. Kerätystä tiedosta voidaan tehdä automaattisesti koneiden tehokkuuslaskelmia, seurata koneiden kulkemia matkoja, tarkastella siirrettyjä massoja ja niin edelleen.

Kaikki kerätty tieto ja ja siitä tehdyt laskelmat auttavat kustannusarvioiden tekemisessä sekä aikataulun arvioimisessa. Kommunikoinnin lisääminen tähtää työmaan tehokkuuden parantamiseen, aikataulujen reaaliaikaiseen seuraamiseen ja ennustamiseen sekä säästöjen tekemiseen.

1.3 Novatron Oy

Novatron Oy on tamperelainen vuonna 1991 perustettu maanrakennuskoneiden mittausjärjestelmiin erikoistunut yritys. Novatron on alallaan Suomen markkinajohtaja sekä yksi Euroopan merkittävimmistä toimijoista. Viimeaikoina kiinnostus Novatronin tuotteisiin on herännyt myös Euroopan ulkopuolella.

Yrityksen päätoimipiste sijaitsee Tampereella. Päätoimipisteestä löytyy hallinnon lisäksi tuotanto, varasto, myynti ja tuotekehitys. Novatronilla on tytäryhtiö Ruotsissa, joka hoitaa Ruotsin myynnin, markkinoinnin sekä tuotetuen. Norjassa on osittain omistettu jälleenmyyjä. Pohjoismaiden ulkopuolella jälleenmyynnistä vastaa saksalainen MOBA AG, joka omistaa Novatronista 25 prosenttia.

Tuotekehityksen, myynnin ja tuotannon ohella yksi tärkeimpiä ja eniten asiakkaalle näkyviä aspekteja on tuotetuki ja huolto. Novatron onkin panostanut tukeen todella paljon. Tukihenkilöt kiertävät aktiivisesti ympäri Suomen hoitamassa laitteiden asennuksia sekä pahimpia ongelmia, joita asiakas ei pysty itse hoitamaan.

Novatron tarjoaa myös asiakkailleen etätukimahdollisuuden, jolla vältetään huoltomiehen käynti työkoneella. Etätuki perustuu langattomaan internet-yhteyteen. Etätuen avulla työkoneen kuljettaja voi lähettää koneelta etätukipyynnön, jolloin Novatronin tuotetuki saa järjestelmän haltuunsa ongelman ratkaisua varten.

Tällä hetkellä Novatronin kehittynein mittausjärjestelmä on *Vision 3D*. Kuvassa 2 Vision 3D on asennettuna työkoneen ohjaamoon. Järjestelmä on asennettavissa erilaisiin työkoneisiin, kuten kaivinkoneisiin, ruoppaajiin ja kaatopaikkajyriin. Järjestelmä koostuu anturoinnista, paikannuksesta sekä ohjaamoon asennettavasta tietokoneesta ja näyttöyksiköstä, jossa mittatieto esitetään työkoneen kuljettajalle kolmiuloitteisena.

Kuva 2: Vision 3D -mittausjärjestelmä

1.4 Työn rakenne

Opinnäytetyö on jaoteltu seuraavalla tavalla. Luvussa 2 esitellään tekniikat, joita harkittiin käytettäväksi järjestelmän pohjana.

Luvussa 3 kerrotaan tarkemmin XMPP -protokollasta, joka valittiin käytettäväksi tekniikaksi. Luvussa käsitellään käyttäjien tunnukset, protokollalaajennokset, XML -viestin pakkaaminen sekä esitellään kaksi palvelintoteutusta.

Luvussa 4 kerrotaan järjestelmän arkkitehtuurista ja siitä, kuinka järjestelmää voidaan käyttää. Luvussa käydään läpi teoriatasolla järjestelmän käyttäminen, varsinaiset esimerkit ja määritelmät ovat liitteinä.

Luvussa 6 tarkastellaan XMPP -protokollan sopivuutta tällaiseen järjestelmään testitulosten valossa sekä todetaan, miten järjestelmän kehityksen kanssa tullaan jatkossa etenemään.

2 Tutkitut tekniikat

Järjestelmän vaatimuksena on alusta asti ollut helppo ja yksinkertainen kommunikaatio monen asiakassovelluksen välillä. Koska kommunikaation vaatimuksena on ollut monen asiakassovelluksen yhtäaikainen ja kaikkien kesken tapahtuva viestien välitys, niin tutkittavien tekniikoiden valitseminen oli yksinkertaista.

Tällä hetkellä on olemassa oikeastaan kaksi täysin avointa protokollaa, IRC ja XMPP, joiden avulla järjestelmä voidaan toteuttaa. Molempien avulla onnistuu ryhmäviestintä ja molemmat ovat yksinkertaisia käyttää.

2.1 IRC

IRC on Jarkko Oikkarisen Oulun yliopistossa 80-luvun lopulla kehittämä avoin tekstipohjainen viestintäprotokolla. Se on pääasiassa tarkoitettu reaaliaikaiseen ryhmäviestintään ja olennainen osa protokollaa on kanavat, jotka toimivat erillisinä keskusteluhuoneina.

Huhtikuussa 2011 sadalla suurimmalla IRC-verkolla oli yhtäaikaa 500000 käyttäjää ja satoja tuhansia kanavia, joita palveltiin noin 1500:lta palvelimelta maailmanlaajuisesti. (Wikipedia)

IRC-verkko koostuu palvelimista ja käyttäjistä. Palvelimet voivat ottaa yhteyden toisiinsa ja näin laajentaa verkkoa. Käyttäjät voivat ottaa yhteyden mihin tahansa yhden verkon palvelimeen ja näin keskustella millä tahansa saman verkon palvelimella olevan käyttäjän kanssa.

IRC:n vahvuus on tekstipohjaisuus ja hyvä skaalautuvuus. Tekstipohjainen kommunikointi ei vie verkkokapasiteettia juuri ollenkaan ja sen takia IRC olisi erittäin sopiva työmaille, joissa työkoneiden järjestelmät ovat yleensä GPRS -yhteyksien varassa.

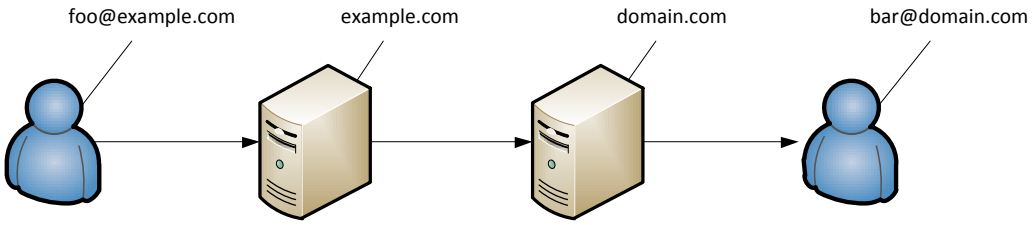
Yksi IRC:n ongelmista on palvelin-palvelin -protokollat, joita on monia erilaisia. Tästä johtuu se, että yleensä IRC -palvelimen voi liittää vain saman palvelinohjelmiston omaavaan palvelimeen. Tämän takia on myös olemassa useita toisistaan eroavia IRC -verkkoja kuten EFnet ja IRCnet.

2.2 XMPP

XMPP on avoimen standardin kommunikointiprotokolla, jonka perustana on XML. XMPP kehitettiin alunperin nimellä Jabber vuonna 1999. Protokolla kehitettiin palvelemaan lähes reaaliaikaista, laajennettavissa olevaa pikaviestintää, tilatietoa ja kontaktilistan hallinointia. Nykyään XMPP:tä voidaan käyttää myös moneen muuhun tarkoitukseen, kuten VoIP ja tiedostojen siirron neuvotteleminen asiakassovellusten välillä.

XMPP:n arkkitehtuuri on samankaltainen IRC:n arkkitehtuurin kanssa. Näiden erona on kuitenkin se, että palvelinten muodostamaa verkkoa ei tarvitse etukäteen määrittää. XMPP -palvelimet osaavat ottaa yhteyden toisiinsa automaattisesti, kun eri palvelimilla sijaitsevat käyttäjät haluavat keskustella keskenään. Tältä osin XMPP muistuttaa todella paljon normaalia sähköpostia.

Kuvassa 3 on kuvattuna tyypillinen tilanne, jossa kaksi eri palvelimilla sijaitsevaa käyttäjää haluavat keskustella keskenään. Kun käyttäjä *foo@example.com* lähettää viestin osoitteeseen *bar@domain.com*, niin palvelin osoitteessa *example.com* osaa välittää viestin palvelimelle *domain.com*, joka välittää viestin edelleen käyttäjälle *bar@domain.com*.



Kuva 3: Käyttäjä *foo@example.com* lähettää viestin käyttäjälle *bar@domain.com*

XMPP:n vahvuuksia ovat hajautettu arkkitehtuuri, avoimet standardit, vankka kehittäjäkunta, turvallisuus ja joustavuus. XMPP pohjautuu XML:ään ja siksi XMPP:n viestit on erittäin vahvasti jäsenneltyjä. XML:stä johtuen sallittujen viestien sisältö voidaan määritellä hyvinkin tarkasti XML-skeemojen avulla.

XMPP:n eräs heikkous on se, että protokollan sisällä ei voi tehokkaasti siirtää binääristä tietoa. Pientä määrää binääritietoa voi kuitenkin kuljettaa esimerkiksi Base64-koodattuna. Esimerkkinä pienestä määrästä dataa joka on vielä järkevissä mitoissa on pikaviestimissä usein käytetyt käyttäjähahmot (ns. avatar, buddy icon).

Suurempien kuormien siirtämiseen täytyy kuitenkin käyttää jotain toista keinoa. Ongelma on jo ratkaistu XMPP:n protokollalaajennoksella, jossa kuvataan tapa neuvotella binääritiedon siirtäminen XMPP -protokollan ulkopuolella.

Toinen XMPP:n ongelmista on verrattaen heikko skaalautuvuus, johtuen muun muassa protokollan XML-pohjaisuudesta. XML:ssä on se ongelma, että palvelin ei etukäteen tiedä, kuinka pitkä asiakkaan lähettämä XML -säkeistö on. XML -virtaa tulee lukea niin kauan, että löydetään säkeistön päättävä XML -elementti. Tämä tarkoittaa sitä, että palvelin joutuu pelkän virran lukemisen lisäksi käsittelemään siinä kulkevan XML:n.

2.3 Vertailu ja yhteenveto

Molemmat protokollat ovat hajautetun luonteensa takia sopivia projektin käyttötarkoitukseen. Molemissa on kuitenkin hyvät ja huonot puolensa, eikä kumpikaan ole selkeästi toista parempi.

Molempien järjestelmien vahvuus on käytännössä myös molempien järjestelmien heikkous. Koska kommunikointi XMPP:llä tapahtuu XML:n avulla, niin se kuluttaa huomattavasti enemmän kaistaa ja vaatii enemmän prosessointitehoa kuin IRC. IRC:ssä taas ei ole mitään standardoitua tapaa lähettää vahvasti jäsenneltyjä viestejä.

Listaus 1: Esimerkki XMPP-viestistä

```
<message from='foo@example.com'
         to='bar@domain.com'>
    <body>
        Hello!
    </body>
</message>
```

Listauksessa 1 on esitetty yksinkertainen XMPP-viesti. Viestin varsinainen sisältö on merkkijono *Hello!*, jonka koko on kuusi tavua, jos oletetaan sen olevan UTF-8 -merkistössä. Viestin koko poislukien turhat välilyönnit ja rivinvaihdot on 82 tavua.

Listaus 2: Esimerkki IRC-viestistä

```
:foo@example.com PRIVMSG bar@domain.com :Hello!
```

Listauksessa 2 on vuorostaan sama yksinkertainen viesti esitettynä IRC -protokollalla. Viestin koko on 47 tavua. Viestien koolla on huima ero. Tässä tapauksessa ero kaventuisi, jos viestin varsinaisen sisällön koko kasvaisi suuremmaksi.

Liitteessä 1 on hieman monimutkaisempi esimerkki XMPP -viestistä. Kuten liitteestä huomataan, niin varsinaisen sisällön lisäksi viestissä on todella paljon itse sisällöstä kertovaa tietoa. XMPP:ssä viestien koko kasvaa nopeasti suureksi. Jos saman viestin sisältämä kommunikointi tehtäisiin IRC:n välityksellä, niin tietoa ei välttämättä muotoiltaisi XML:llä vaan jollain tiiviimmällä esitystavalla.

Järjestelmän kannalta ryhmäviestintä on erittäin oleellinen ominaisuus. XMPP:ssä on mahdollista toteuttaa tällainen joko MUC- tai PubSub -protokollan avulla. IRC:ssä ryhmäviestintä on sisäänrakennettu.

XMPP ja IRC tekevät myös eron siihen, kuinka keskusteluhuoneet on hajautettu. XMPP:ssä huoneet ovat aina yhdellä palvelimella, kun taas IRC:ssä huoneet ovat hajautettu koko verkon laajuudelle. XMPP:lle on kuitenkin kehitysasteella oleva DMUC -protokollalaajennos, joka sallisi keskusteluhuoneiden hajauttamisen.

Projektin puitteissa päädyttiin valitsemaan näistä kahdesta protokollasta XMPP. Valinnan kriteereinä oli molempien tekniikoiden heikkoudet ja vahvuudet, valmiit palvelin- ja asiakasohjelmatoteukset, laajennettavuus sekä dokumentaatio.

XMPP tarjoaa suhteellisen valmiin pohjan järjestelmän toteuttamiselle verrattuna IRC:n. XMPP:lle löytyy monissa ympäristöissä toimivia ja helppokäyttöisiä palvelimia sekä asiakassovelluksia. Parhaillaan XMPP -palvelimen saa asennettua toimintakuntoon alle kymmenessä minuutissa. Tietysti tarkempi konfigurointi ja optimointi vie enemmän aikaa.

XMPP:n päälle oli helppo lähteä tekemään uutta kommunikointiprotokollaa XML:n avulla. Helppous syntyy siitä, että XMPP -yhteisössä on vuosien saatossa muodostunut hyvin määritellyt prosessit alkuperäisen protokollan laajentamiseen. Ottamalla mallia olemassa olevista laajennoksista saa ideoita omiin kokeiluihin.

XMPP:n dokumentaatio on erittäin hyvää ja selkeää. Tämä johtuu tietysti suurilta osin siitä, että XMPP on IETF:n standardoima protokolla. Standardien lisäksi XMPP:n ympärillä toimiva yhteisö on todella aktiivinen ja laaja. Aktiivinen yhteisö kertoo protokollasta, joka ei tule kuolemaan lähitulevaisuudessa.

3 XMPP

3.1 JID -tunnus

Jokaisella XMPP -entiteetillä on JID -tunnus. XMPP -entiteetti on mikä tahansa järjestelmän osa, joka voi kommunikoida XMPP -protokollan avulla. Esimerkiksi palvelimet ja asiakassovellukset ovat XMPP -entiteettejä.

JID -tunnus koostuu paikallisesta osasta, verkkotunnusosasta sekä resurssiosasta. Esimerkiksi:

Listaus 3: JID -tunnus

```
oskari.timperi@example.com/oskari
```

jossa paikallinen osa on *oskari.timperi*, verkkotunnusosa *example.com* ja resurssi *oskari*. JID muistuttaa hieman sähköpostiosoitetta. Käyttäjien lisäksi myös palveluilla ja palvelimilla on JID, tosin niiden JID koostuu yleensä vain domain-osasta.

JID voi olla joko paljas tai täydellinen. Paljaalla JID:llä tarkoitetaan JID:ä joka on muotoa *oskari.timperi@example.com* tai *example.com* palvelimen tapauksessa. Täydellisellä JID:llä tarkoitetaan JID:ä joka on muotoa *oskari.timperi@example.com/koti* tai *example.com/resurssi* palvelimen tapauksessa.

JID -tunnuksen tarkka määrittely on RFC 6122 -dokumentissa.

3.2 XEP

Yksi XMPP:n parhaita puolia on sen helppo laajennettavuus. XMPP -protokollaa onkin vuosien saatossa laajennettu ns. XEP:llä. Näitä laajennoksia on tehty useaan eri tarkoitukseen, kuten paikkatiedon jakamiseen, tiedostojen jakamiseen, asiakassovellusten ominaisuuksien tiedusteluun ja niin edelleen.

XEP:n voi periaatteessa tehdä kuka tahansa. Jos on tarpeeksi hyvä idea ja sitä lähtee toteuttamaan XEP -prosessin mukaisesti, niin todennäköisesti laajennos lisätään viralliselle hyväksyttyjen laajennosten listalle. XEP -listaa ylläpitää XMPP Standards Foundation.

3.2.1 XEP-0045: Multi User Chat

XMPP ei itsessään toteuta IRC:n perusominaisuutta, eli ryhmäkeskustelua. Tätä varten on kehitetty laajennosprotokolla, jossa määritellään kuinka tällainen ominaisuus saadaan käyttöön myös XMPP:ssä. Laajennos vaatii palvelimelta tuen laajennokselle, jotta viestien välitys keskustelijoiden välillä toimisi.

MUC -palvelun käyttö on kehitetty hyvin samankaltaiseksi kuin normaalin XMPP:nkin. Yksi lähtökohta on ollut se, että keskusteluhuoneiden käyttäminen tulisi olla mahdollista, vaikka asiakasohjelmalla ei varsinaisesti MUC -palvelulle tukea olisikaan. Tämän takia huoneiden JID:it ovat samanlaisia, kuin normaalien käyttäjienkin.

3.3 Palvelimet

XMPP:n avoimen luonteen takia sitä tukevia palvelinohjelmistoja on saatavilla paljon. Erona palvelimissa on yleensä lisenssi, ohjelmointikieli ja tuetut laajennokset.

Yksi mielenkiintoinen palvelin on *ejabberd*, joka on ohjelmoitu *Erlang* -ohjelmointikielellä. Erlang on on Ericssonin kehittämä ohjelmointikieli, joka on tarkoitettu hajautettujen ja vikasietoisten sovellusten kehittämiseen.

ejabberd:ssä on Erlangin takia suora tuki palvelinsovelluksen hajauttamiselle useammalle palvelintietokoneelle. Hajauttamisen seuraus on se, että jos yksi palvelin hajoaa, niin se ei vaikuta palvelinsovelluksen toimintaan. Lisäksi palvelinsovelluksen suorituskyky paranee, koska käyttäjät voidaan jakaa useammalle fyysiselle palvelimelle.

Toinen suosittu palvelinsovellus on *OpenFire*. OpenFire on ohjelmoitu Javalla ja siihenkin on saatavilla tuki hajauttamiselle, mutta sovellus ei tue sitä automaattisesti. OpenFireen on helppo ohjelmoida omia liitännäisiä ja sitä on todella helppo käyttää WWW -pohjaisen käyttöliittymän kautta.

Tässä projektissa tullaan pääasiassa käyttämään OpenFire -palvelinta, koska se on Novatronilla jo ennestään tuttu yrityksen sisäisessä pikaviestinnässä. Tässä opinnäytetyössä ei oteta tarkemmin kantaa palvelinsovellusten asentamiseen tai käyttöön.

3.4 Viestien pakkaus

XMPP -palvelimet toteuttavat yleensä XML -virran pakkauksen. Jos asiakasohjelma tukee pakkausta, niin asiakkaan ja palvelimen välillä käytävän keskustelun käyttämää kaistaa pystytään hieman pienentämään. Tässä täytyy kuitenkin ottaa myös huomioon lisääntynyt prosessoinnin tarve sekä palvelimen, että asiakkaan päässä. Palvelimelta tämä voi teoriassa vaatia paljonkin prosessoriaikaa, jos siihen on yhdistynyt paljon asiakkaita.

Asiakkaana taas voi olla esimerkiksi älypuhelin, ja pakkauksen purkaminen vaikuttaa lisääntyneen prosessoinnin seurauksena akun kestoon. Yleensä pakkauksen aiheuttamaa ylimääräistä prosessointia kuitenkin pidetään pienempänä pahana kuin suurempaa verkkokuormaa.

Toinen mahdollisuus minimoida viestin koko on binääri-XML. Binääri-XML on tapa esittää XML tiiviissä binäärimuodossa. Binääri-XML -muotoja on useita, esimerkiksi W3C:n kehittämä EXI -formaatti, jota pidetään varteenotettavimpana binääri-XML -määrittelynä (Wikipedia: Efficient XML Interchange).

Efficient XML Interchange -työryhmä on tehnyt tutkimuksen, jossa verrataan muutamia binääri-XML muotoja, XML+GZIP -yhdistelmää sekä normaalia XML:ää. Tutkimuksessa on mitattu binäärimuotojen nopeutta ja tiivistämistä. Tutkimuksen mukaan binääri-XML:llä voidaan saavuttaa 70% tiiviimpiä dokumentteja ilman XML -skeeman apua. Skeeman avulla on mahdollista päästä noin 80%:n tiivistykseen. Nopeus on jaettu XML -dokumentin tuottamiseen ja dokumentin käsittelemiseen. Binääri-XML:llä saavutettiin 50% nopeampi dokumentin tuottaminen ja 180% nopeampi käsittely verrattuna normaaliin XML:ään. (White, Kangasharju, Brutzman & Williams 2007)

3.5 MUC -palvelun skaalautuvuus

XMPP:n MUC -palvelu ei toistaiseksi skaalaudu kovin hyvin tilanteeseen, jolloin kanavalle lähetetää nopeasti viestejä monen keskustelijan toimesta. Esimerkkinä voidaan pitää MUC -palvelun kanavaa, jolla on N keskustelijaa. Oletetaan, että nämä N keskustelijaa lähettävät kanavalle sekunnin välein viestin.

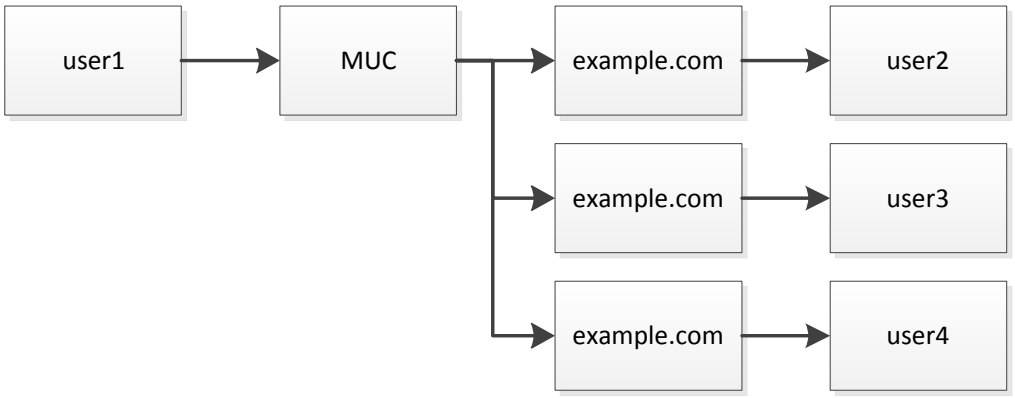
Kun yksi keskustelija lähettää kanavalle viestin, niin palvelin välittää viestin N:lle keskustelijalle (ml. viestin lähettäjä). Kun N keskustelijaa tekee saman yhtäaikaisesti, niin palvelin välittää jokaisen keskustelijan viestin N keskustelijalle.

Palvelin vastaanottaa N viestiä ja lähettää $N * N$ viestiä kerran sekunnissa. Tästä voidaan listauksen 1 mukaisen minimaalisen XMPP -viestin koon (82 tavua) perusteella arvioida tietomäärää, joka palvelimen tulee käsitellä joka sekunti.

Jos $N = 10$, niin palvelin vastaanottaa $N * 82 = 10 * 82 = 820$ tavua ja lähettää $N * N * 82 = 100 * 82 = 8200$ tavua joka sekunti. Jos $N = 100$, niin vastaavat luvut ovat 8200 ja 820000.

Tilanne pahenee siis nopeasti kanavalla olevien keskustelijoiden määrän ja lähetettävän tiedon mukaan. Viestien määrä, jotka palvelin välittää, on suoraan verrannollinen käyttäjien määrän toiseen potenssiin. Kymmenen vastaanotettua viestiä tarkoittaa sataa lähetettyä viestiä. Sadalla viestillä palvelin lähettää kymmenentuhatta viestiä.

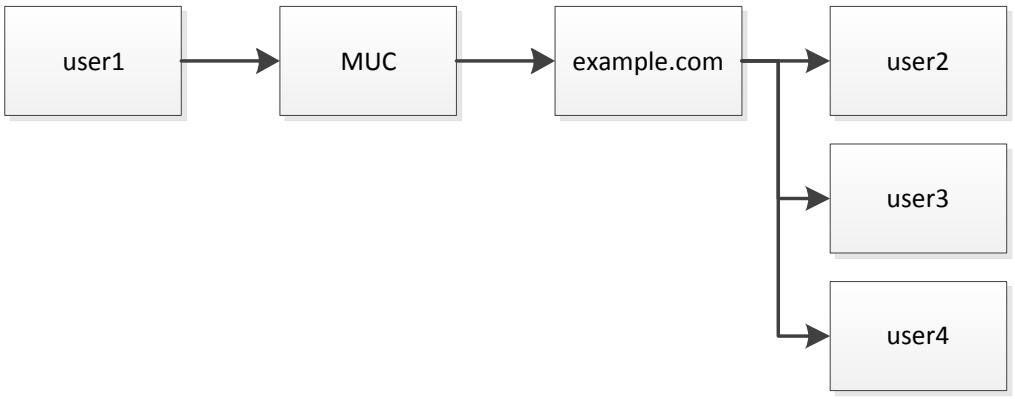
Yksinkertaisuuden vuoksi ylläoleva kuvaus ei täysin kuvaa tämänhetkistä MUC -palvelun toimintaa. Tällä hetkellä palvelu toimii kuvan 4 mukaan.



Kuva 4: MUC -palvelun toiminta tällä hetkellä

Käytännössä palvelin siis ottaa vastaan N kappaletta viestejä ja välittää eteenpäin N kappaletta viestejä. Tämä pahentaa palvelimen kuormaa entisestään, koska palvelimen täytyy käsitellä jokainen sille lähetetty viesti.

Kuvassa 5 on kuvattu optimitilanne MUC:n toiminnasta. Kuvan 5 tilanne on lähempänä tilannetta, joka kuvattiin yllä.

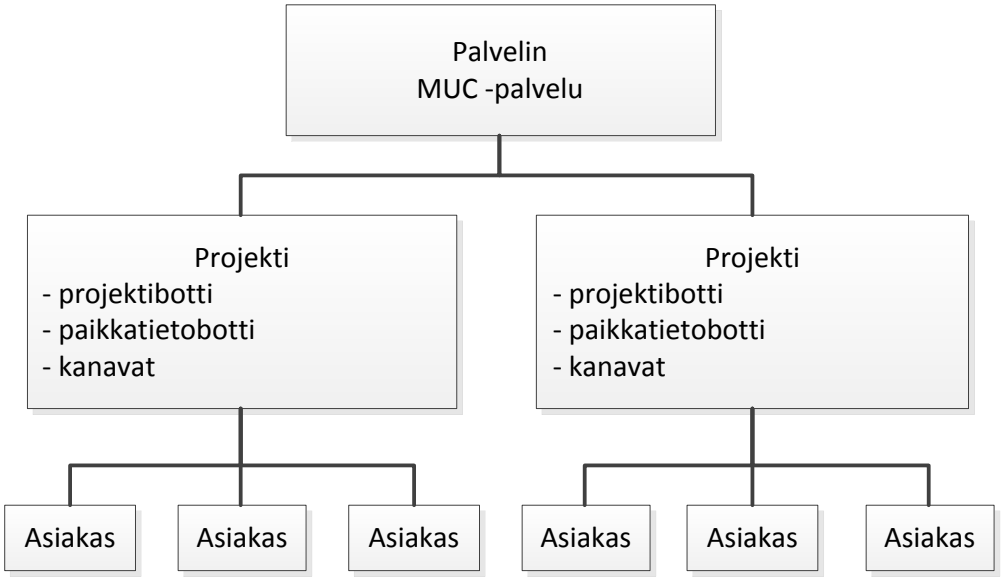


Kuva 5: Kuinka MUC -palvelun tulisi toimia

Tässä tapauksessa palvelin tutkisi vain yhden viestin ja tekisi tarvittavat toimenpiteet, jotta viesti kulkisi edelleen vastaanottajille. Kuvan 5 kaltainen toiminto vaatisi palvelimelta sille eksplisiittisen tuen.

4 Arkkitehtuuri

4.1 Yleiskuvaus

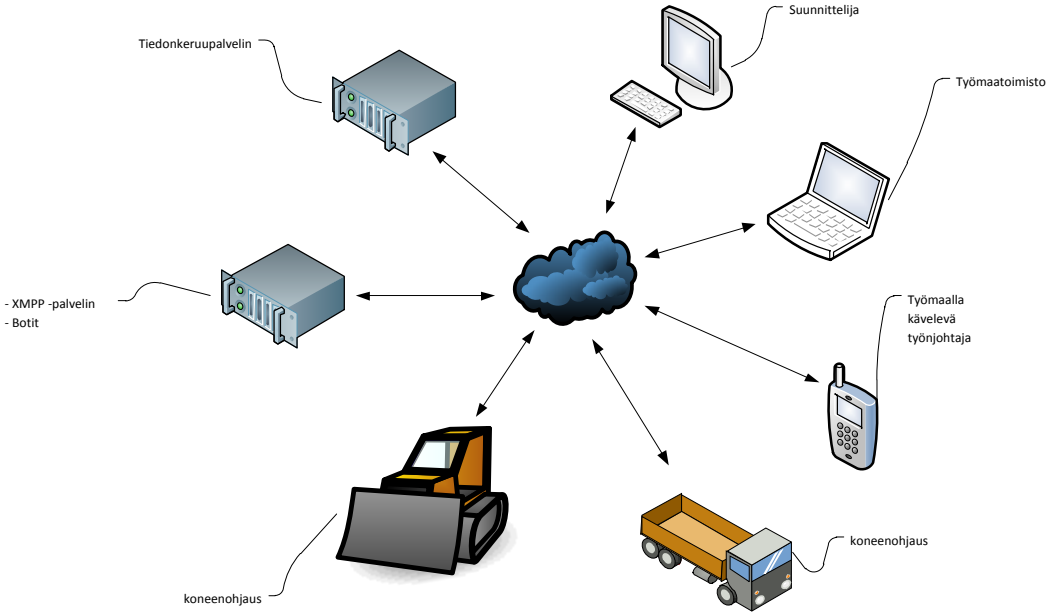


Kuva 6: Arkkitehtuuri

Kuvan 6 mukaisesti järjestelmän perustana on XMPP -palvelin ja siinä toimiva MUC -palvelu. Seuraavalla tasolla on kuvattuna projekti, joka käytännössä koostuu erilaisita boteista ja niiden hallinnoimista MUC -kanavista. Botti voi olla tavallinen asiakassovellus, jolle on toteutettu jonkinlainen toiminnallisuus.

Koska projektien toiminnallisuus päädyttiin toteuttamaan bottien ja MUC -kanavien avulla, niin käytettävällä palvelinohjelmistolla ei ole väliä. Ainut vaatimus palvelimelle on se, että se toteuttaa MUC -palvelun. Järjestelmän pohjalla on asiakas, joka voi olla mitä tahansa kaivinkoneen ja työmaasuunnittelijan väliltä.

Yksinkertaisimmassa järjestelmässä yksi fyysinen palvelin on yhden projektin käytössä. Kuvassa 7 on kuvattu järjestelmä jossa projektille on varattu yksi fyysinen palvelin.



Kuva 7: Yksinkertainen järjestelmä

Tällä palvelimella ovat ajossa XMPP -palvelin, projektibotti sekä lokaatiobotti. Työmaan tilaa seuraava tiedonkeruusovellus toimii omalla palvelimellaan. Tiedonkeruusovellus suhtautuu projektiin samalla tavalla, kuin esimerkiksi työkoneessa oleva asiakassovellus.

Järjestelmästä voi tehdä niin monimutkaisen kuin XMPP ja MUC vain sallivat. Esimerkiksi yhdellä fyysisellä palvelimella sijaitseva XMPP -palvelin, jossa on useita MUC -palveluita joissa jokaisessa on useita projekteja. Tällainen järjestelmä vaatii palvelimelta paljon enemmän, kuin edellä kuvattu yksinkertainen tapaus. Parempi tapa onkin hajottaa järjestelmä pienempiin yksiköihin.

Kun järjestelmän yhdistää pilvipalveluiden kanssa saadaan todella mielenkiintoinen asetelma. Tulevaisuudessa voidaan esimerkiksi nappia painamalla laukaista uusi instanssi Amazonin EC2 -pilvipalvelusta, johon on asennettu valmiiksi kaikki tarvittavat sovellukset. Tämän jälkeen kysytään projektin tiedot ja projekti käynnistetään.

4.2 Botit

4.3 Projektibotti

Projektin ydin on projektibotti, joka vastaa yhden projektin hallinnoimisesta. Projektiin kuuluu joukko kanavia, asiakassovelluksia sekä metadataa. Projektiin rekisteröityminen ja projektin yleisen tiedon saanti hoituu projektibotin kautta.

Projektibotti antaa asiakkaalle oikeuden liittyä projektin kanaville, asiakkaan suoritettua onnistuneen rekisteröitymisen. Toistaiseksi rekisteröitymiseen riittää pelkkä salasana, jolla ei ole monimutkaisuusvaatimuksia.

Projektin salasanan ei tarvitse välttämättä olla vaikea, jos halutaan että esimerkiksi työkoneen kuljettaja voi työmaalla tarvittaessa antaa salasanan. Salasana voidaan myös jättää määrittämättä, jolloin projektiin voi liittyä kuka tahansa ilman salasanaa. Tällöin tulee pitää huoli siitä, että työmaa on vain testikäytössä tai muuten turvattu.

Projektibotin toiminnallisuus on pidetty hyvinkin yksinkertaisena. Projektibotin kanssa käytävästä kommunikoinnista kerrotaan enemmän luvussa 5.

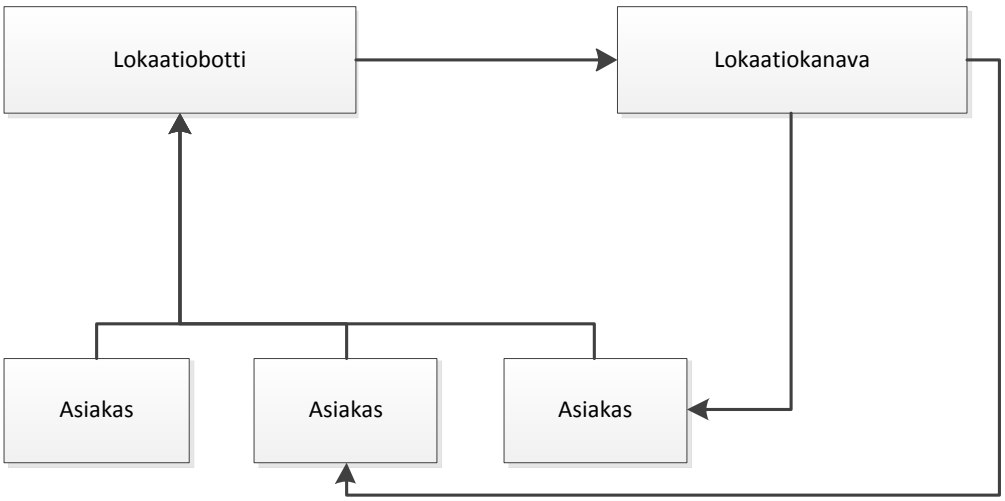
4.4 Lokaatiobotti

Lokaatiobotin tarkoituksena on ottaa vastaan paikkatietoa asiakkailta, jotka sitä haluavat jakaa. Lokaatiobotti koostaa ja jakaa paikkatiedon paikkatietokanaville, joita muut asiakkaat voivat kuunnella.

Paikkatieto välitetään botin kautta, koska suoraan kanavalle lähetettäessä tulee nopeasti palvelimen prosessointikyky vastaan. Kuten kappaleessa 3.5 todettiin, nopean tiedon jakaminen kanavan kautta vie todella paljon resursseja.

Bottia käytettäessä palvelimen kuormitus paikkatiedon osalta kasvaa lineaarisesti paikkatietoa *kuuntelevien* asiakkaiden osalta. Voi olla tilanteita, joissa paikkatietoa jakava asiakas ei ole kiinnostunut muiden paikkatiedosta, jolloin kyseisen asiakkaan ei tarvitse olla paikkatietokanavalla kuuntelemassa sitä. Näin meneteltäessä voidaan pienentää palvelimen kuormaa.

Kuvassa 8 on esitetty lokaatiobotin toiminta-ajatus.



Kuva 8: Lokaatiobotin toiminta

4.5 Kanavat

Projektiin liittyy tietty joukko kanavia, joita pitkin projektiin liittyvää tietoa jaetaan. Jokaisella kanavalla on oma tarkoituksensa ja niitä tulee käyttää sen mukaisesti. Tämän työn puitteissa toteutettiin vain kaksi kanavaa, joiden avulla projektiin liittyneiden asiakkaiden kesken voidaan vaihtaa tila- sekä paikkatietoa.

Aula
Aulakanavan tarkoitus on pitää kirjaa projektiin jollakin ajanhetkellä liittyneistä asiakkaista. Aulan tarkoituksena on tarjota projektiin liittyneille asiakkaille tilatietoa muista asiakkaista.

Paikkatieto
Paikkatietokanavan tarkoitus on jakaa paikkatietoa sitä haluavien asiakkaiden välillä.

4.6 Asiakas

Järjestelmä tarjoaa tavan jakaa tietoa. Jotta järjestelmästä olisi hyötyä, tarvitaan asiakkaita jotka jakavat, keräävät ja analysoivat tietoa.

Järjestelmän näkökulmasta asiakas voi olla mitä tahansa. Asiakas voi olla työkone, joka lähettää omaa paikkatietoaan järjestelmään. Toinen asiakas voi kerätä tätä tietoa ja tehdä sen pohjalta analyysejä. Analyysien pohjalta voidaan laatia esimerkiksi Internet-sivuja, joilla on kuvaajia työn etenemisestä ynnä muusta.

5 Kommunikaatio

Projektin kanssa kommunikointi on loppujen lopuksi hyvin yksinkertaista. Periaatteessa se onnistuu millä tahansa XMPP -asiakassovelluksella. Kommunikointi on tehty valmiita XMPP -protokollamäärittelyjä ja laajennoksia silmälläpitäen siten, että olemassaolevia tekniikoita on pyritty käyttämään mahdollisimman paljon sellaisenaan.

Tässä kappaleessa on kuvattu tärkeimmät osa-alueet projektin kommunikaatiosta sekä esitetty havainnollistavia esimerkkejä. Kommunikaatioprotokollan XML-skeemat löytyvät liitteestä 4. Protokollan skeemat täydentävät XMPP -protokollaa ja hyödyntävät myös XMPP:n laajennusprotokollia.

5.1 Projektiin liittyminen

Projektiin liittyminen tapahtuu ottamalla yhteys projektibottiin. Ensimmäinen asia joka botin kanssa täytyy hoitaa, on tunnistautuminen. Projektille on luontivaiheessa määritelty salasana, joka tulee tunnistautumisvaiheessa antaa projektibotille.

Tunnistautumisprosessi alkaa siitä, kun asiakas lähettää projektibotille rekisteröitymisviestin. Projektibotti vastaa kysymällä projektin salasanaa. Asiakkaan vastatessa oikealla salasanalla, projektibotti lisää asiakkaan projektin kanavien jäsenlistoille ja toivottavaa asiakkaan tervetulleeksi. Liitteessä 2 on esimerkki asiakkaan ja projektibotin välisestä keskustelusta rekisteröitymisen aikana.

Hyväksytyn tunnistautumisen jälkeen asiakas voi kysyä projektibotilta projektista tarkempia tietoja. Asiakas voi myös liittyä projektin kanaville. Projektin kanavat asiakas voi kysyä projektibotilta.

5.2 Projektin tiedot

Projektille on määritelty tietty minimimäärä tietoa, joka sen täytyy asiakkaille tarjota. Näihin lukeutuu projektin koordinaatit, projektin omistaja, projektin kuvaus ja niin edelleen.

Minimitietojen lisäksi projekteille voi määritellä projektikohtaisia tietoja. Projektikohtaisia tietoja ja tiedon määrää ei ole rajoitettu. Projektikohtaisiin tietoihin voidaan tallentaa esimerkiksi metadataa työmaan suunnittelutyökaluista.

Liitteessä 3 on annettu esimerkki projektin tietojen sekä projektiin liittyvien kanavien kysymisestä sekä projektibotin vastauksesta kyselyihin.

5.3 Paikkatieto

Paikkatiedon jakaminen tapahtuu jo aiemmin kappaleessa 4.4 kuvatulla tavalla. Paikkatietoa jakava asiakas lähettää tietonsa lokaatiobotille, joka välittää paikkatiedon edelleen paikkatietokanaville.

Paikkatietoa kuunteleva asiakas voi liittyä paikkatietokanaville, jonne lokaatiobotti välittää sille lähetetyt paikkatiedot. Paikkatietokanavia voi olla useita riippuen siitä, kuinka usein kanavaa päivitetään. Näin esimerkiksi raportteja laativat asiakkaat voivat kuunnella kymmenen minuutin välein päivittyvää paikkatietokanavaa ja reaaliaikaisempaa tietoa vaativat asiakkaat voivat kuunnella puolen minuutin välein päivittyvää tietoa.

Liitteessä 1 on esimerkki paikkatiedon lähettämisestä lokaatiobotille.

6 Yhteenveto

Opinnäytetyön aiheena oli suunnitella kommunikointiprotokolla ja toteuttaa sitä käyttävä järjestelmä rakennustyömaita silmälläpitäen. Tarkoituksena oli luoda helposti laajennettavissa oleva protokolla ja siinä onnistuttiin hyvin jo pelkästään teknologiavalinnalla. Opinnäytetyön puitteissa protokollaan suunniteltiin vain siltä vaadittavat perusominaisuudet.

Protokollan suunnittelu ja toteuttaminen eivät olleet yksinkertaisia asioita. Suunniteltaessa tuli miettiä tarkkaan eri vaihtoehtoja ja keskustella kolleegojen kanssa niistä. Suunnittelua helpotti kuitenkin se, että keskityttiin vain protokollan ytimeen ja perusominaisuuksiin.

Järjestelmän kehitys ei kuitenkaan lopu tähän opinnäytetyöhön. Jatkossa järjestelmä tullaan integroimaan Novatronin tuotteisiin. Näin muut voivat hyödyntää tietoa, jota Novatronin järjestelmistä voidaan työmailla kerätä.

Toivottavaa jatkoa ajatellen on myös se, että järjestelmästä saataisiin muilta organisaatioilta palautetta. Jos järjestelmä todetaan liian tehottomaksi, niin näiden kokemusten pohjalta voidaan kehittää XMPP -protokollaa ja palvelimia tehokkaammiksi esimerkiksi Binääri-XML:ää hyödyntäen. Vaihtoehtoisesti voidaan myös tutkia muita tapoja toteuttaa yhteinen kommunikointijärjestelmä.

LÄHTEET

InfraTM TEKES loppuraportti 2010. Luettu 10.10.2011.
http://www.rts.fi/infrabim/InfraTM_pilotti_Tampere_Oulu_loppuraportti.pdf

Kilpeläinen 2007. Tietomallipohjainen automaatio tieverkon päällystämisen korjaus- ja uudisrakentamistyössä (TIMARA). VTT-R-09689-07.

Infra - Rakentaminen ja palvelut 2001-2005, Teknologiaohjelmaraportti 4/2006, Loppuraportti. 2006. TEKES. Helsinki.

Peter Saint-Andre 2011. Request For Comments: 6122. ISSN 2070-1721.
http://tools.ietf.org/html/rfc6122

Peter Saint-Andre 1999. XEP-0045: Multi-User Chat. Luettu 10.10.2011. Päivitetty 16.7.2008.
http://xmpp.org/extensions/xep-0239.html

White, Kangasharju, Brutzman & Williams 2007. Efficient XML Interchange Measurements Note. Luettu 10.10.2011.
http://www.w3.org/TR/2007/WD-exi-measurements-20070725/

Wikipedia. IRC. Päivitetty 7.12.2011. Luettu 9.12.2011.
http://en.wikipedia.org/wiki/Irc

Wikipedia. Efficient XML Interchange. Päivitetty 6.12.2011. Luettu 9.12.2011.
http://en.wikipedia.org/wiki/Efficient_XML_Interchange

LIITE 1

Paikkatiedon lähettäminen lokaatiobotille

```
<message
  from='abcdef@example.com/abcdef'
  to='jid@example.com/res'
  id='geo6'>
  <x xmlns='urn:xmpp:trak:events'>
    <events>
      <movementevent>
        <object
            id='my-excavator'
            type='excavator'/>
        <object
            id='pekka'
            type='user'/>
        <location>
          <geoloc
              xmlns='http://jabber.org/protocol/geoloc'>
            <lat>62.70124</lat>
            <lon>24.59871</lon>
          </geoloc>
        </location>
      </movementevent>
    </events>
  </x>
</message>
```

Rekisteröityminen projektiin

```
<iq from='abcdef@example.com/abcdef'
    id='reg1'
    to='project-master@example.com'
    type='get'>
  <query xmlns='jabber:iq:register'/>
</iq>

<iq from='project-master@example.com'
    id='reg1'
    to='abcdef@example.com/abcdef'
    type='result'>
  <query xmlns='jabber:iq:register'>
    <password/>
  </query>
</iq>

<iq from='abcdef@example.com/abcdef'
    id='reg2'
    to='project-master@example.com'
    type='set'>
  <query xmlns='jabber:iq:register'>
    <password>secret</password>
  </query>
</iq>
```

Asiakas antoi virheellisen salasanan

```
<iq from='project-master@example.com'
    id='reg2'
    to='abcdef@example.com/abcdef'
    type='error'>
    <error type='modify'>
        <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    </error>
</iq>
```

Salasana hyväksyttiin

```
<iq from='project-master@example.com'
    id='reg2'
    to='abcdef@example.com/abcdef'
    type='result'/>

<message
    from='project-lobby@example.com'
    to='abcdef@example.com'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <invite from='project-master@example.com' />
  </x>
</message>
```

Asiakas kysyy projektin tietoja

```
<iq from='abcdef@example.com/abcdef'
    id='proj1'
    to='project-master@example.com/bot'
    type='get'>
  <query xmlns='urn:xmpp:trak:info'/>
</iq>

<iq from='project-master@example.com'
    id='proj1'
    to='abcdef@example.com/abcdef'
    type='result'>
    <query xmlns='urn:xmpp:trak:info'>
        <created>TIMESTAMP</created>
        <title>TITLE</title>
        <owner>OWNER</owner>
        <x xmlns='jabber:x:data' type='result'>
            <field var='FORM_TYPE' type='hidden'>
                <value>urn:xmpp:trak:project:info</value>
            </field>
            <field type='text-single' var='some-variable'>
                <value>some value</value>
            </field>
            <field type='text-single' var='project-id-for-some-tool'>
                <value>id</value>
            </field>
        </x>
    </query>
</iq>
```

LIITE 3

Asiakas kysyy projektiin liittyvät kanavat

```
<iq from='abcdef@example.com/abcdef'
    id='proj1'
    to='project-master@example.com/bot'
    type='get'>
  <query xmlns='urn:xmpp:trak:channels'/>
</iq>

<iq from='project-master@example.com'
    id='proj1'
    to='abcdef@example.com/abcdef'
    type='result'>
    <query xmlns='urn:xmpp:trak:channels'>
        <channel jid='project-lobby@example.com' type='lobby' />
        <channel jid='project-location@example.com' type='location'
    />
    </query>
</iq>
```

events.xsd

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:trak:events'
  xmlns='urn:xmpp:trak:events'
  elementFormDefault='qualified'>

  <xs:import
    namespace='urn:xmpp:trak:location'
    schemaLocation='http://novatron.fi/schemas/trak/location.xsd'/>

  <xs:annotation>
    <xs:documentation>
      Schema for a protocol to communicate state change events across
     XMPP.
    </xs:documentation>
  </xs:annotation>

  <xs:element name='x'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='events'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='events' type='Events' />

  <xs:complexType name='Events'>
    <xs:sequence>
      <xs:choice minOccurs='0' maxOccurs='unbounded'>
        <xs:element name='event' type='Event'/>
        <xs:element name='movementevent' type='MovementEvent'/>
        <xs:element name='stateevent' type='StateEvent'/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='Event'>
    <xs:sequence>
      <xs:element name='object' type='Object' maxOccur='unbounded'/>
    </xs:sequence>
    <xs:attribute name='id' type='xs:string' use='required'/>
    <xs:attribute name='serial' type='xs:string' use='optional'/>
    <xs:attribute name='timestamp' type='xs:string' use='optional'/>
    <xs:attribute name='from' type='xs:string' use='optional'/>
  </xs:complexType>
```

```
<xs:complexType name='MovementEvent' xmlns:loc='urn:xmpp:trak:
 location'>
  <xs:complexContent>
    <xs:extension base='Event'>
      <xs:sequence>
        <xs:choice minOccur='0'>
          <xs:element name='transform' type='Transform'/>
          <xs:element ref='loc:location'/>
        </xs:choice>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name='StateEvent'>
  <xs:complexContent>
    <xs:extension base='Event'>
      <xs:sequence>
        <xs:element name='state' type='State' minOccur='0'/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name='Object'>
    <xs:attribute name='id' type='xs:string' use='required'/>
    <xs:attribute name='type' type='xs:string'/>
</xs:complexType>

<xs:complexType name='Transform'>
  <xs:sequence>
    <xs:element name='translate' type='Vector3' minOccur='0'/>
    <xs:element name='rotate' type='Quaternion' minOccur='0'/>
    <xs:element name='scale' type='Vector3' minOccur='0'/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name='State'>
  <xs:sequence>
    <xs:choice minOccur='0'>
      <xs:element name='color' type='Color'/>
      <xs:element name='colorrgba' type='ColorRGBA'/>
    </xs:choice>
    <xs:element name='visibility' type='Visibility' minOccur='0'/>
    <xs:element name='highlight' type='Hightlight' minOccur='0'/>
  </xs:sequence>
</xs:complexType>
```

```
  <xs:complexType name='Visibility'>
      <xs:attribute name='value' type='xs:boolean'/>
  </xs:complexType>

  <xs:complexType name='Highlight'>
      <xs:attribute name='value' type='xs:boolean'/>
  </xs:complexType>

  <xs:complexType name='ColorRGBA'>
    <xs:attribute name='r' type='colorValueType'/>
    <xs:attribute name='g' type='colorValueType'/>
    <xs:attribute name='b' type='colorValueType'/>
    <xs:attribute name='a' type='colorValueType'/>
  </xs:complexType>

  <xs:simpleType name='Color'>
    <xs:restriction base='xs:decimal'>
      <xs:minInclusive value='0.0'/>
      <xs:maxInclusive value='1.0'/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name='Quaternion'>
    <xs:attribute name='w' type='xs:decimal'/>
    <xs:attribute name='x' type='xs:decimal'/>
    <xs:attribute name='y' type='xs:decimal'/>
    <xs:attribute name='z' type='xs:decimal'/>
  </xs:complexType>

  <xs:complexType name='Vector3'>
    <xs:attribute name='x' type='xs:decimal'/>
    <xs:attribute name='y' type='xs:decimal'/>
    <xs:attribute name='z' type='xs:decimal'/>
  </xs:complexType>
</xs:schema>
```

location.xsd

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:trak:location'
  xmlns='urn:xmpp:trak:location'
  elementFormDefault='qualified'>

  <xs:import
    namespace='http://jabber.org/protocol/geoloc'
    schemaLocation='http://www.xmpp.org/schemas/geoloc.xsd'/>

  <xs:annotation>
    <xs:documentation>
      Schema for an extended geoloc element.
    </xs:documentation>
  </xs:annotation>

  <xs:element name='location' xmlns:geoloc='http://jabber.org/
    protocol/geoloc'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='radius' type='nonNegativeInteger' minOccur
    ='0'/>
        <xs:element name='position' minOccur='0'>
          <xs:complexType>
            <xs:attribute name='x' type='double'/>
            <xs:attribute name='y' type='double'/>
            <xs:attribute name='z' type='double'/>
          </xs:complexType>
        </xs:element>
        <xs:element ref='geoloc:geoloc' minOccur='0'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

project_info.xsd

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:trak:info'
  xmlns='urn:xmpp:trak:info'
  elementFormDefault='qualified'>

  <xs:import
    namespace='jabber:x:data'
    schemaLocation='http://www.xmpp.org/schemas/x-data.xsd'/>

  <xs:import
    namespace='urn:xmpp:trak:location'
    schemaLocation='http://novatron.fi/schemas/trak/location.xsd'/>

  <xs:annotation>
    <xs:documentation>
      Schema for a query to get projects detailed information such as
     when the
      project was created and what is the bounding box of the project
    . One
      can include project specific information in the optional data
    form.
    </xs:documentation>
  </xs:annotation>

  <xs:element name='query'>
    <xs:complexType
        xmlns:xdata='jabber:x:data'
        xmlns:loc='urn:xmpp:trak:location'>
      <xs:sequence minOccurs='0'>
        <xs:element name='created' type='xs:dateTime'/>
        <xs:element name='title' type='xs:string'/>
        <xs:element name='owner' type='xs:string'/>
        <xs:element ref='loc:location'/>
        <xs:element ref='xdata:x' minOccurs='0'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

project_channels.xsd

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:trak:channels'
  xmlns='urn:xmpp:trak:channels'
  elementFormDefault='qualified'>

  <xs:element name='query'>
    <xs:sequence>
      <xs:element name='channel' type='ProjectChannel'/>
    </xs:sequence>
  </xs:element>

  <xs:complexType name='ProjectChannel'>
    <xs:attribute name='jid' type='xs:string'/>
    <xs:attribute name='type' type='xs:string'>
      <xs:simpleType>
        <xs:restriction base='xs:string'>
          <xs:enumeration value='lobby'/>
          <xs:enumeration value='location'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:schema>
```